BS-pFL: Enabling Low-Cost Personalized Federated Learning by Exploring Weight Gradient Sparsity

Lening Wang
ECE Department
University of Houston
Houston, USA
lwang56@uh.edu

Manojna Sistla
ECE Department
University of Houston
Houston, USA
msistla@central.uh.edu

Mingsong Chen
Department of Embedded Software and Systems
East China Normal University
Shanghai, China
mschen@sei.ecnu.edu.cn

Xin Fu
ECE Department
University of Houston
Houston, USA
xfu8@central.uh.edu

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892137

***Abstract*—Recent advancements in Convolution Neural Networks (CNNs) have achieved amazing success in numerous applications. The record-breaking performance of CNNs is usually at the prohibitive training costs, thus all training data are usually processed at the powerful centralized server side, which rises privacy concerns. Federated learning (FL) is a distributed machine learning method over mobile devices to train a global model while keeping decentralized data on devices to preserve the data privacy. However, there are two major limitations to deploy FL on mobile clients. Firstly, on the client side, the limited communication and computation resources on mobile devices cannot well support the full training iterations. Secondly, on the server side, conventional FL only aggregate a common output for all the clients without personalizing the model to each client, which is an important missing feature when clients have heterogeneous data distributions. In this work, we aim to enable low-cost personalized FL by focusing on the weight gradients which are the most important exchanging parameters in FL and meanwhile, dominating the computation and communication cost. We first observe that the client's calculated weight gradients have high sparsity, and the sparse pattern in weight gradients could be predicted via very simple bit-wise operations on a sequence of bits (named bit-stream) instead of conducting expensive high-precision calculations to determine them. Furthermore, a unique pattern is exhibited in each client's uploaded weight gradients according to the distribution of its local training data. Guided by this pattern, each client can get a personalized aggregated model to fit its own data. Hence, we leverage bit-streams to predict weight gradients sparsity for low-cost training on each device, and meanwhile, bit-streams are used to represent the unique sparse pattern of the weight gradient for each client which will guide the model personalization. From our experiments, our proposed framework can improve the computation efficiency by 3.5× on average (up to 4.2×) and reduce the communication cost by 23% on average (up to 41%) while still achieving the state-of-the-art personalized accuracy.**

***Index Terms*—Personalized Federated Learning**

I. Introduction

Deep learning based AI technology, especially, Convolution Neural Networks (CNNs) have achieved amazing success in numerous applications in recent years [16], [27]. However, the record-breaking performance of CNNs is at the prohibitive training cost due to the required massive training data and iterations. Usually, training data are collected from different users (i.e. clients) but training tasks are performed at the powerful centralized server side, which rises data privacy concerns. Federated learning (FL) is a distributed machine learning paradigm which allows each client training the network using its own data, and the centralized server only aggregates those models to obtain one final global model. Hence, local data will be kept on each client to preserve the data privacy. However, there are two major challenges to deploy FL on mobile clients. Firstly, at the client side, unlike the powerful centralized server, the clients are usually mobile devices with limited computation and communication resources, it is very challenging for them to perform the complex training iterations which are needed for each training example. This challenge is even exacerbated as the CNN models are becoming deeper and the structures are becoming more complex. Secondly, at the server side, conventional FL only develops a common generalized model for all the clients without personalizing the model to each client, which is an important missing feature when clients have heterogeneous data distributions.

In this study, we aim to address these two challenges and explore the low-cost personalized federated learning by focusing on the weight gradients which are the most important exchanging parameters in FL and meanwhile, dominating the computation and communication cost during the model training in mobile devices. Firstly, on each client, we observe high sparsity exists in the weight gradients calculated by each training example. Skipping the near-zero weight gradient could dramatically reduce the training cost for FL without hurting the accuracy. However, we need to first conduct expensive high-precision calculations on each weight gradient to identify near-zero gradients, and then skipping them could achieve very limited time and energy saving. Fortunately, we find that the weight gradient significance could be predicted without calculating. This is because each weight gradient is a sum of products (named partial sums) between forward activations and back propagated errors, and the magnitude of one weight gradient is highly related to the number of its significant partial sums. In this work, we define significant weight gradients/partial sums are those with large magnitudes. As can be seen, one can predict the significance of a weight gradient by simply counting the number of its significant partial sums. Secondly, we further observe high sparsity exists in the weight gradients uploaded by each client during aggregations, and each client has its own unique sparse pattern in the uploaded weight gradients according to its local training data. In other

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

words, the distribution of each client's local training data can be well represented by that pattern. As a result, guided by the sparse pattern, we can generate a personalized aggregated model for each client to fit its own local data.

We propose a low-cost **p**ersonalized **FL** framework (BS-pFL) through exploring two levels of Bit-Streams: (1) Bit-Stream at the weight gradient level for each training example in a client (named $BS_{example}$) to predict the significant weight gradients and significant partial sums, and (2) Bit-Stream at the client level (named BS_{sync}) that represents the sparse pattern of weight gradients uploaded to the server. $BS_{example}$ will guide the client for substantial computation and communication skipping on weight gradients for **low-cost** FL. Especially, for each weight gradient, we transfer its corresponding activations and errors into bit-streams, and perform the simple bit-wise operations to identify the significance in a lightweight way without the expensive calculations. Only the predicted significant weight gradients and partial sums are calculated in full precision, while others are either pruned or quantized, therefore, the computation costs could be significantly reduced. The communication costs are also reduced by skipping uploading pruned weight gradients. Furthermore, a **personalized** model will be generated at the server side for each client according to its BS_{sync}. From our experiments, our proposed framework can improve the computation efficiency by 3.5× on average (up to 4.2×) and reduce the communication cost by 23% on average (up to 41%) while still achieving the state-of-the-art personalized accuracy.

II. Related Work

Sparsity in Neural Networks: Network sparsity has been widely studied and adopted as an acceleration technique that can be applied during both the training and inference of neural networks [2], [9], [19], [25]. For example, [24] has proposed to prune the activation gradients during the back propagation. [10] permanently removed the weights that have negligible contributions towards the outputs and produced excellent model compression. In a more extensive pruning techniques such as [12], [15], [17], [21], [22], a complete channel or filter is pruned to give better compression and uniform structure to the network. Lottery Tickets [9] prunes 20% of the network every 50,000 training iterations and could get 5–10× model size reduction. Weight gradients pruning is proposed for training acceleration by reducing communication cost of weight gradients exchanging in distributed learning system. [1] prunes 99% weight gradients with the smallest absolute value by a heuristic algorithm. But those pruned weight gradients are still determined through calculation.

Personalized Federated Learning: The data heterogeneity in federated learning makes it hard to learn a single shared global model for all individual clients. To address this challenges, personalized Federated Learning (pFL) is proposed to personalize the global model for each client in the federation. Several different sets of techniques are proposed to implement pFL. one set of techniques treat pFL as a model agnostic meta-learning (MAML) problem [7], [8], [13], [28]. However, MAML approaches will introduce tremendous computation cost as they rely on the Hessian matrix. Another set of techniques learn a mixture of the global and local models [3], [5]. For example, [11] introduced a new mixed model that the model is divided into the global base layers and on top personalized layers. In pFedMe [6], a method uses Moreau envelops as the client regularized loss to decouple the personalized and global model optimizations. Finally, some techniques learn from "closer" clients to implement pFL. The clients are "closer" than others in terms of data distribution. In [18], the close clients are grouped together and a centralized model is trained per group. In pFedFOMO [26], each client only federates with a subset of relevant clients.

III. Background and Optimization Goals

Fig. 1(a) shows the process of FL, a centralized server will distribute an initialized CNN model to selected mobile clients (step ①), who will then train the model using their own data in step ② and ③. Table I shows the definitions and notations in our BS-pFL, similar to regular CNN training, there are four stages involved in training convolution ($Conv$) layers on each mobile device: $Forward\ Pass$ (FP): O_l = $I * W_l$; $Back\ Propagation$ (BP): $\delta_l = W_l^T * \delta_{l+1}$; $Weight\ Gradients\ Calculation$ (WGC): $\Delta W_l = \delta_{l+1} * I_l$; and $Weight\ Updating$ (WU): $W_l = W_l - \eta \cdot \Delta W_l$. After a predefined interval, which is defined as one communication round, each mobile client will send its calculated weight gradients back to the server in step ④. In conventional FL, in step ⑤, the sever then will aggregate the weight gradients from all the clients to produce one global model. This process repeats until the training process finished. However, in pFL, the server will generate a unique model for each client in step ⑤ to fit its own data.

Optimization Goals: Similar to FL, pFL can be divided into two stages: the regular training on mobile clients using the their local data (step ② to ③) and the personalized aggregation on the server side using uploaded weight gradients (step ⑤). Each stage has different optimization goals. In the first stage (i.e., step ② to ③), the main goal is to reduce the computation cost on the client side without sacrificing the training accuracy. Hence, we propose to use $BS_{example}$ in step ② and ③ to predict the significant weight gradients and significant partial sums and prune/quantize the insignificant ones. In the second stage (step ⑤, the main goal is to aggregate personalized models on the server side for individual clients. We propose to use BS_{sync} in step ⑤ to represent the sparse pattern of weight gradients, and meanwhile achieving personalized aggregation Overall, We propose BS-pFL to achieve low-cost personalized federated learning.

IV. Reduce Computation on Individual Clients

A. Motivations and Challenges

Existing works [4], [14], [20], [24] show that there are numerous insignificant parameters in neural networks, removing them during the calculation stage can effectively reduce

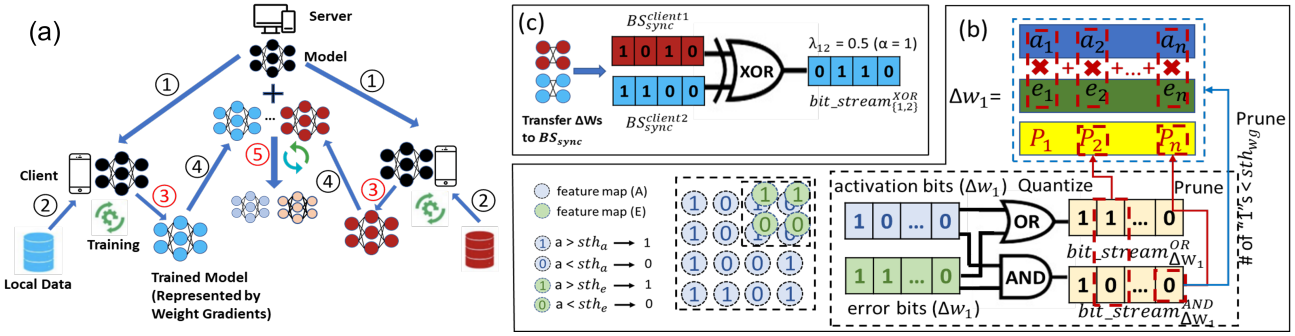

Fig. 1. Overview of BS-pFL. (a) Training Flow in BS-pFL; (b) Weight Gradient Significance Prediction based Pruning and Quantization on the Clients; (c) Personalized Aggregation on the Server.

Notation	Definitions
I_l	input of layer l at $Forward\ Pass$
O_l	output of layer l at $Forward\ Pass$
W_l	weight of $Conv$ layer l
δ_{l+1}	layer $l+1$'s back-propagated error, the gradient of O_l
ΔW_l	weight gradient of $Conv$ layer l
$*$	2-D convolution
η	learning rate
W_l^T	reversed matrix of W_l
$BS_{example}$	Bit-Stream at the weight gradient level for each training example
BS_{sync}	Bit-Stream that represents the sparse pattern of weight gradients
bit_stream	reference bit streams produced according to corresponding operations
$p_{wg}\%$	pruned percentage of weight gradients
$p_{ps}\%$	pruned percentage of partial sums

TABLE I
DEFINITIONS AND NOTATIONS

computation complexity without sacrificing accuracy. On each client, we observe that during the WU stage, many weight gradients' absolute values are very small. It is intuitive that skipping the updatings of those close-to-zero weight gradients (defined as insignificant weight gradients) will not affect the training accuracy. Furthermore, the convolution operations can be decomposed into a series of Multiply–Accumulate (MAC) operations, thus, the remaining significant weight gradients can be represented as the summation of multiple products between forward activations and back propagated errors. We define a single product as a partial sum for a certain weight gradient. It is well-known there are a large amount of extremely close-to-zero parameters in side CNNs, For example, there are a large amount of extremely close-to-zero values in the δ_ls and I_ls, leading to many zero or close-to-zero partial sums (defined as insignificant partial sums) in those significant weight gradients. It is reasonable that the WGC stage will not be impacted by pruning those insignificant partial sums. Since most computation operations during WGC stage are the massive high-precision MAC operations, we propose to skip MAC operations related to insignificant weight gradients and insignificant partial sums in significant weight gradients, therefore, efficiently accelerating the training process. For simplicity purposes, we define these two kinds of sparsity (i.e., insignificant weight gradients, and insignificant partial sums in significant weight gradients) as the gradient sparsity.

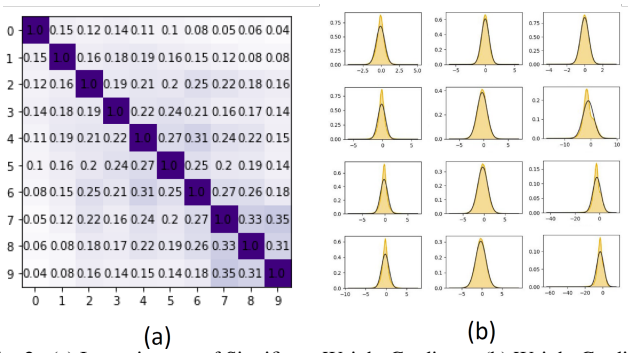

Fig. 2. (a) Inconsistency of Significant Weight Gradients; (b) Weight Gradient Distribution of VGG-16 During Sparse Training.

However, we are facing two major challenges to leverage the gradient sparsity for training acceleration. The significance of each weight gradient varies across the training process. Fig. 2 (a) shows the consistency of weight gradients' significance during the training process of VGG-16 on Cifar100. We select the largest 20% (in absolute value) weight gradients from 10 different training periods and record their overlapping in percentages from every two training periods. The horizon axis and vertical axis represent different training periods. As the figure shows, on average, there are only 18% weight gradients remaining significant in any two different training periods, which implies the low consistency of weight gradients' significance across the training procedure. Therefore, the one-time significant identification and one-time pruning schemes used in weight sparsity cannot be naturally adapted to gradient sparsity. The significance identification per weight gradient needs to be conducted at each training example. The second challenge is that the insignificant weight gradients are irrelevant to the weight sparsity. In other words, one cannot use insignificant weights to predict the insignificant weights gradients. Fig. 2 (b) presents the weight gradient distribution of VGG-16 during training when a weight pruning method is applied. The columns represent different selected $Conv$ layers from VGG-16, while the rows represent weight distributions sampled at different training periods (from the beginning to the end). We prune 20% of the network weights every

100,000 training iterations, thus the model's sparsity iteratively increases from row 1 to row 4. As it shows, with near-zero weights are pruned iteratively during the training, there is no noticeable changing in the distribution of weight gradients and most of them are still located at the near-zero range in all $Conv$ layers. This implies the insignificant weight gradients cannot be identified according to the weights, they need to be calculated first for the identification. Since weight gradients are produced at every training example and the major delay and energy bottleneck in WGC is the high-cost computation on MAC operations to obtain weight gradients, identifying and pruning the insignificant weight gradients and partial sums after that expensive computation have trivial improvement on training efficiency.

B. Observations and Implementation

To address the aforementioned challenges, we first made two key observations related to gradient sparsity: 1) In FL, insignificant weight gradients and partial sums could be pruned without affecting training accuracy, the remaining significant weight gradients could be further approximated via quantizing the middle-range partial sums. 2) the significance of a weight gradient is related to the number of significant partial sums it contains. Then we implement a light-weight Bit-stream based Significance Prediction Scheme (BSPS) to locate the insignificant weight gradients and partial sums.

1) Observation Validation: Observation 1: Table II verifies observation 1. We train three networks (LeNet (MNIST), VGG-16 (Cifar10), and ResNet18 (Cifar10)) with 10 clients: we adopt four kinds of pruning/quantizing techniques: a) pruning smallest [1] $p_{wg}\%$ of all the weight gradients; b) pruning the smallest $p_{ps}\%$ partial sums in each weight gradient; c) a combination of a) and b); and d) quantizing mid-range partial sums based on c). Table II shows our experimental results. We observe that network achieves a relatively high accuracy even 50% of the weight gradients are pruned while comparing to the baseline (i.e., the model obtained through regular FL). For the partial sums, we have similar observation that the network maintain comparable accuracy when 30% of the partial sums are pruned. Our results prove that both the insignificant weight gradients and partial sums could be pruned without sacrificing the training quality. While combining a) and b) together, we find weight gradient pruning and partial sum pruning can be conducted concurrently without causing further accuracy drop. This indicates that the significant partial sums in significant weight gradients are the main contributors in FL training. We also observe that the significant weight gradients could be further approximated by quantizing some mid-range partial sums: with smallest 30% of partial sums and 50% of weight gradients pruned, we further randomly quantize another 30% partial sums in mid-range (the partial sums with their absolute values between smallest 30% and largest 30% partial sums) to 8bit FP. No accuracy drop was observed. The accuracy results are shown in the last column of Table II. All these results show we can speed up the training process by first pruning the insignificant weight gradients, then prune the insignificant partials sums and quantize the mid-ranged partial sums in significant gradients.

Observation 2: We investigate the relationship between weight gradient's significance and the number of partial sums by analyzing the weight gradients on clients with various datasets. We record each weight gradient with the number of significant partial sums it contains. The top 50% (measured in absolute value) of the partial sums from each layer are defined as significant partial sums. For illustration purpose, we pick a typical client as an example, and Fig. 3 depicts each weight gradient from VGG-16 in that client by plotting the number of significant partial sums it contains at the horizon axis and its significance at the vertical axis. The significance of a weight gradient is calculated by normalizing its absolute value to (0, 1). The horizon axis is also normalized to (0, 1) because the number of partial sums is different in different layers. As Fig. 3 shows, there is a linear relationship between the significance of a weight gradient and the number of significant partial sums it contains. Hence we can use the number of significant partial sums to predict the significance of weight gradients. Since each partial sum is the product between an activation and an error. Large activations and errors will produce large partial sums. Instead of actually calculating the real partial sums, we can obtain their significance from their corresponding activations and errors.

2) Methodology: Achieving Low-Cost Weight Gradient Approximation via Bit-stream based Significance Prediction Scheme (BSPS): Innovated by these observations, we propose the bit-stream based significance prediction method to predict all the significant/insignificant partial sums and weight gradients.

Locating Significant Partial Sums and Weight Gradients: The significant partial sum is defined as a partial sum with large absolute value, which is the product between a large absolute-valued activation and a large absolute-valued activation error. Fig. 1 (b) depicts our bit-stream based insignificance prediction for partial sums and weight gradients. As it shows, for each weight gradient, its I_l and δ_{l+1} are transferred into bit-streams where activations/errors over a threshold are marked as 1's while others are 0's. These on-client bit streams are produced for each training example, thus named as $BS_{example}$. By conducting AND logic operations between the activation bit stream and the corresponding error bit stream, a reference bit stream (i.e., $bit\_stream_i^{AND}$) is produced. "0" in the stream implies a small product while "1" indicates a relatively large product, which is the predicted significant partial sums. For example, P_1 (marked as"1") in Fig. 1 plays a more important role to Δw_1 comparing to P_2 which is marked as "0". As shown in Observation 2, more significant partial sums a weight gradient contains, the higher significance it exhibits. We then can predict the significance of a weight gradient via counting the number of 1's in the reference bit stream. A weight gradient with higher number (above a certain threshold) of 1's will have higher probability being

[1] This paper uses absolute values for comparisons.

kept for further computing, while a small number implies an insignificant weight gradient whose computation can be skipped. We use Distribution Based Threshold Determination (DBTD) scheme introduced in [23], [24] to set the thresholds according to the pruning rate, which is determined through our experiences.

Locating Insignificant and Middle-Range Partial Sums: With the identification and pruning of insignificant weight gradients, we then predict and prune/quantize the insignificant and middle-range partial sums inside the remaining significant weight gradients. The insignificant partial sums are defined as products between close-to-zero activations and close-to-zero errors. To predict insignificant partial sums, we regenerate a new bit stream $bit\_stream_i^{OR}$ for each significant weight gradient via conducting OR operations between its activation and error bit-streams. 0's in $bit\_stream_i^{OR}$ indicate the insignificant products while 1's in the aforementioned $bit\_stream_i^{AND}$ represent the significant products, and the remaining products are at the significant-insignificant margin, named as mid-range partial sums. We employ three different approximation levels to partial sums which have different contributions to the significant weight gradient: (1) eliminating the insignificant partial sums; (2) high-precision computing for the significant partial sums; and (3) quantizing computing for the partial sums that are in the significant-insignificant margin. Note that the activations produced during FP stage are stored in memory for future use in WGC stage, the insignificant activations (marked as 0's in the activation bit-stream) are stored in a low-precision format to relief both the memory access and computation stress.

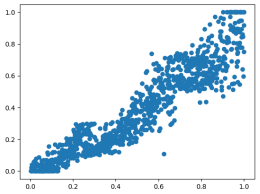

Fig. 3. Relationship Between Weight Gradient Significance and the Number of Significant Partial Sums.

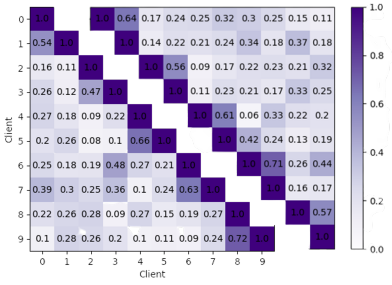

Fig. 4. Similarity of Uploaded Significant Weight Gradients between Any Two Clients

V. Personalized Aggregation on Server

A. Motivation and Challenge

Conventional FL outputs a global generalized model on the server side by aggregating the weight gradients from all clients without considering each client's personalization, which is an important missing feature when given clients have the heterogeneous data distributions. For each client, a personalized aggregated model is necessary to fit its own data. In FL, a individual client cannot be generalized well according to the limited local training data. For pFL, a client should learn more from adjacent clients with similar data distribution but ignore the knowledge from those clients with unrelated datasets. Thus, each client has unique contributions to other clients based on the similarity of their training data. This motivates us to aggregate a personalized model for each client by using a unique contribution coefficient λ_{mn} to estimate the contribution between any two mobile clients m and n. The λ_{mn} is defined as the similarity of local data distribution between client m and n. However, the main challenge is each client's local data is protected and cannot be accessed by others, hence, preventing us from getting λ_{mn} directly.

B. Observations and Implementation

To get λ_{mn}, we observe each client has its own unique sparse pattern in their uploaded weight gradients according to its local training data, and the distribution of each client's local training data can be reflected by that pattern.

1) Observation Validation: We train two networks, LeNet (MNIST) (in upper right of Fig. 4) and ResNet18 (Cifar10) (in bottom left of Fig. 4) with 10 clients. We adopt the learning setup and the evaluation protocol described in [6] to generate heterogeneous clients in terms of the classes and sizes of local training data. Two clients (e.g., $client_0$ and $client_1$, $client_2$ and $client_3$, and etc.,) share the training data with similar distributions. We select the largest 25% (in absolute value) weight gradients from each client and plot the overlapping in percentages between any two clients in Fig 4. The horizon axis and vertical axis represent different mobile clients. As the figure shows, on average, there are about 61% weight gradients are simultaneously selected in two different clients sharing similar training data, while only 18% weight gradients are simultaneously selected in any two different clients with different training data.

2) Methodology: Achieving Personalized Model Aggregation via Exploring Similarity of Uploaded Bit Streams: Innovated by above observation, we propose using weight gradient bit-stream to determine the value of λ_{mn}. As shown in Fig. 1(c), We transfer the uploaded weight gradients into bit-streams (named BS_{sync}) where the uploaded weight gradients over a threshold are marked as 1s while others are marked as 0s. This transferring process can be done at the server side. By conducting XOR logic operations between two BS_{sync}s from $client_m$ and $client_n$, a reference bit stream (i.e., $bit\_stream_{mn}^{XOR}$) is produced. Then λ_{mn} is determined according to Eq. 1. Note here, λ_{mn} is clipped to 1 and α ($\geq$ 1) is a pre-defined coefficient represents the trade-off between personalization and generalization. A larger α will set λ_{mn} to 1, which is the same as conventional FL for a generalized common model.

$$\lambda_{mn} = \alpha * \frac{number\ of\ "0"s\ in\ bit\_stream_{mn}^{XOR}}{length\ of\ bit\_stream_{mn}^{XOR}} \quad (1)$$

For each personalized model w^n for $client_n$, we have:

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta \sum_{m=1}^{N} \frac{\lambda_{mn}}{\sum_{m=1}^{N} \lambda_{mn}} \nabla_{\mathbf{w}^m} \quad (2)$$

Where η stands for learning rate and N stands for total number of clients.

Network	Baseline	(a) Prune p_{wg}% Weight Gradients			(b) Prune p_{ps}% Partial Sums			(c) Prune Both	(d) Quantizing
		p_{wg}=50	p_{wg}=60	p_{wg}=70	p_{ps}=30	p_{ps}=40	p_{ps}=50	p_{wg},p_{ps}=50,30	
LeNet	95.90%	95.70%	95.67%	95.60%	95.82%	95.65%	95.57%	95.73%	95.75%
VGG-16	83.02%	82.83%	82.56%	81.89%	82.85%	82.42%	80.72%	82.32%	82.27%
ResNet18	84.03%	83.67%	83.51%	82.15%	83.57%	83.52%	81.41%	83.55%	83.48%

TABLE II
ACCURACY VALIDATION ON WEIGHT GRADIENT PRUNING AND QUANTIZATION

VI. BS-PFL

Algorithm 1 shows the algorithm of BS-pFL. BS-pFL optimized the FL from two aspects: 1)BS-pFL achieves low-cost computation on mobile clients by skipping insignificant weight gradients and partial sums; 2)BS-pFL aggregates a personalized model at the server side for each client.

Algorithm 1: BS-pFL Algorithm

Input: Learning Rate: η, Communication Round: T, Initialize Model: w
Output: w_1 to w_n
procedure Client-Side Optimization
for *each communication round* $t \in T$ **do**
  for *each client n in parallel* **do**
    forall Δw_n **do**
      Significance Prediction by BSPS
      if *predict* Δw_n *is small* **then**
        prune the current Δw_n
      end
      else
        prune the insignificant partial sums
        quantize mid-ranged partial sums
        calculate significant partial sums
        calculate Δw_n
      end
    end
  end
end
procedure Server-Side Aggregation
calculate λ_{mn} **based on** $bit\_stream_{mn}^{XOR}$
return personalized w_n **according to Eq 2**

VII. EVALUATION

A. Experimental Setup

Model Selection and Dataset: We adopt four different neural network models from light-weight to large-scale in our experiments: LeNet, AlexNet, VGG-16 and ResNet18. We evaluate our pFL training framework on two different image classification tasks: MNIST and CIFAR10. LeNET is trained on MNIST and Cifar-10; others are trained on Cifar-10. For each dataset, we adopt [6] for generating heterogeneous clients in terms of classes and sizes of local training data. On each client, examples are randomly split into training set (80%) and testing set (20%), with the same distribution.

Compared Methods: We evaluate and compare the following approaches: (1) BS-pFL, our proposed pFL training framework; (2) BS-FL, our proposed pFL framework without personalized aggregation; (3) FedAvg, one of the most widely used FL algorithms; (4) FedAvg-FT, fine-tune the model with local data at last round; (5) FedPer, a pFL approach that learns per-client personal classifier on top of a shared feature extractor. We record the average test accuracy of all local models for evaluation on their own testing set.

Method	MNIST	Cifar-10			
	LeNet	LeNet	AlexNet	VGG-16	ResNet-18
FedAVG	95.45%	40.17%	50.11%	82.19%	83.01%
FedAVG-FT	97.65%	43.52%	61.45%	85.92%	86.21%
FedPer	97.57%	44.24%	61.22%	85.02%	86.45%
BS-FL	95.40%	40.15%	51.02%	82.11%	82.55%
BS-pFL	97.45%	44.44%	61.07%	84.87%	86.23%

TABLE III
ACCURACY RESULTS OVER 10 CLIENTS ON MNIST AND CIFAR-10

Implementation: All the experiments are conducted using tensorflow framework. We replace the auto-differentiation technique in tensorflow by customized back-propagation methods. Models are using the $Categorical-cross-entropy$ loss function. The calculated weight gradients are updated with the help of $Adam$ optimizer function, with default settings of $beta\_1 = 0.9, beta\_2 = 0.999, epsilon = 1e-07$. The $learning\ rate$ has been appropriately selected. We set p_{wg}%=50%, p_{ps}%=30% for all benchmarks if not noted. The personalized aggregation is performed at last two communication rounds to ensure convergence. The local epoch is set to 20 and the maximum communication round is set to 100. We simulate a centralized server with 10 clients on our workstation (Intel-Xeon E5-2650 CPU and Tesla P100 GPUs).

Threshold Determination: At the client side, two thresholds (sth_a and sth_e) are used to generate the bit-streams from activation and error maps, another two (sth_{wg} and sth_{ps}) are used to determine the percentage of significant weight gradients to be calculated and the percentage of partial sums to be pruned. The pruning rate of insignificant weight gradients and partial sums are selected manually based on experimental experiences. Once the pruning rate is determined, sth_{ps} is determined as the bit-stream length is determined. The values of sth_{wg} can be calculated by DBTD proposed in [23], [24]. Then sth_a and sth_e are adjusted automatically during training. If too many partial sums are defined as insignificant partial sums based on our desired pruning rate, the values of sth_a and sth_e decrease and vise versa. Therefore, only two hyperparameters need to be predefined (pruning rate for weight gradients and partial sums) on the client side while the thresholds can be auto-tuned. On the server side, we use a threshold $sth_{uploded}$ to transfer the uploaded weight gradients into corresponding bit streams, which is set to select the largest 25% weight gradients.

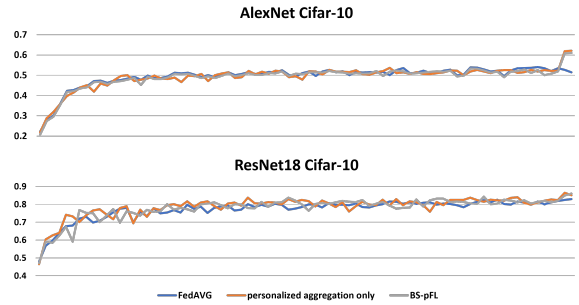

Fig. 5. Accuracy Curves on Different Datasets

B. Experimental Results and Analysis

1) Accuracy Validation and Model Convergence: **Accuracy Improvement through Personalized Aggregation:** The accuracy results are shown in Table. III. Our BS-pFL shows better accuracy on personalized tasks than conventional FL framework FedAVG. While comparing to personalized FL framework, our BS-pFL maintains comparable accuracy but can achieve dramatically computation and communication reductions. Note here, although BS-pFL is designed for effective personalized FL, it can also be applied to conventional FL for generalized aggregation by just conducting the low-cost weight gradient approximation on individual clients, which actually becomes BS-FL.

Convergence Guarantee: To validate the convergence of BS-pFL, we compare the training performance of FedAVG, BS-pFL and our scheme without on-client weight gradient approximation(i.e. personalized aggregation only). We track the learning process from selected models of all communication rounds. Fig. 5 shows our personalized aggregation scheme converges to higher accuracy compared to FedAVG after personalized steps. Fig. 5 also shows the learning curve for BS-pFL is similar to the one with personalized aggregation only, proving that our proposed algorithm at the client side has negligible impact on the convergence of the network.

2) Computation Reduction: To measure the impact of our proposed algorithm on the computational cost, we collect the number of floating point operations (FLOPs) per epoch required on Cifar-10 and MNIST of the conventional parameter-based pFL method (e.g. FedAVG-FT) and our BS-pFL method. Then, the computation reduction is evaluated in the reduction of FLOPs, which is a universal metric for all platforms. The computation reductions obtained by BS-pFL over the conventional pFL method are listed in Table IV where LeNeT is trained on MNIST while others are trained on Cifar-10. As it shows, our BS-pFL has reduced the computational cost up to $4.2\times$ and $3.5\times$ on average. The FLOPs reductions are similar across different networks because they are determined by the desired pruning rate. Since we use a similar pruning rate ($p_{wg}\%$=50%, $p_{ps}\%$=30%) for all benchmarks, the reductions are similar as well. For LeNet, we use a much aggressive pruning rate $p_{wg}\%$=70%, $p_{ps}\%$=50%, hence we can get more FLOPs reductions. Our bit-stream based schemes only use 1-bit to record the significance and performs bit-wise operations, thus the overhead is negligible to the overall training FLOPs.

Network	conventional pFL	BS-pFL	Computation Reduction
LeNet	$1.44T$	$343G$	$4.19\times$
AlexNet	$2.58T$	$775G$	$3.32\times$
VGG-16	$56.1T$	$16.9T$	$3.31\times$
ResNet-18	$7.28T$	$2.19T$	$3.31\times$

TABLE IV
COMPUTATION REDUCTION BY FLOPS COUNT

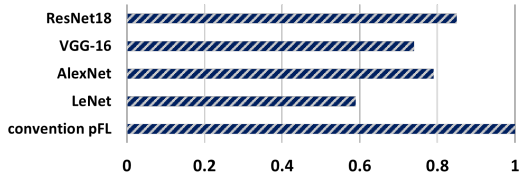

Fig. 6. Communication Efficiency

3) Communication Reduction: We evaluate the communication overhead in the size of the uploaded data, results are shown in Fig. 6. Compared with conventional pFL methods, BS-pFL can achieve communication reduction by around 23% through pruning the uploaded weight gradients.

C. Sensitivity Analysis on Hyperparameters

Table. V shows the accuracy on different selections of hyperparameters.

Impact of α: As mentioned in Eq. 2, α controls the trade-off between personalization and generalization ability. For example, larger value of α means that the knowledge coefficient should approach to 1, and more generalization ability should be gained. Table. V shows the results of BS-pFL with different αs. In most settings a significantly large value α will hurt the performance of BS-pFL. In this study, we set α as 1.5.

Impact of Personalized Step R: R stands for the number of our personalized aggregation rounds performed at the end of FL communication rounds. We compare the performance of BS-pFL under different setting of personalize steps (e.g., R = 1, 2, 3, ∞). ∞ means the personalized aggregation is performed from the first communication round. The results in Table. V show that larger value of R cannot always lead to better performance, or even cause convergence issues. Usually, the personalized aggregation step can help to improve the knowledge transfer between clients when the clients are sharing similar models. In this study, we set R as 2.

Impact of Local Epochs E: E stands for the number of epochs trained on aggregation rounds. We select different values of E to see the impact on accuracy, demonstrated in Table. V. The results show that a relatively larger value of E can improve the performance of personalized aggregation. Hence we chose E = 30 in aggregation rounds.

VIII. DISCUSSION AND FUTURE WORK

The Pruning Rate Determination for BS-pFL. We determine the pruning rate through our experiences in this study. However, we also find the pruning rate could be varies from layers and different training epochs. How to dynamically tune the thresholds is remained in our future work.**The hardware support for BS-pFL.** It is hard to transfer potential FLOPs

Dataset	Network	α				R				E			
		1	1.5	4	8	1	2	8	∞	10	20	30	40
MNIST	LeNet	97.44%	97.45%	96.92%	96.02%	97.47%	97.45%	96.42%	92.12%	96.11%	97.41%	97.45%	97.46%
Cifar-10	LeNet	44.42%	44.44%	43.85%	41.57%	44.22%	44.44%	40.29%	38.19%	41.21%	44.22%	44.44%	44.42%
	AlexNet	61.09%	61.07%	60.98%	54.88%	61.17%	61.07%	56.57%	49.99%	55.17%	57.45%	61.07%	61.05%
	VGG-16	83.92%	84.87%	82.25%	82.18%	84.77%	84.87%	81.15%	76.45%	82.44%	84.09%	84.87%	84.89%
	ResNet-18	86.22%	86.23%	85.45%	83.11%	86.21%	86.23%	83.15%	75.44%	83.11%	85.51%	86.23%	86.24%

TABLE V
ACCURACY RESULTS ON DIFFERENT SETTINGS OF HYPERPARAMETERS

reductions into actual speedup while using our BS-pFL on current computing platforms. We need extra hardware support to generate the reference bits and perform bit-wise operations in a low-cost and high-efficient way. We leave the design of specific hardware as our future work.

IX. CONCLUSION

In this work, we study the weight gradient related sparsity during the training process of FL. We found the sparsity can not only be leveraged to reduce the computation cost of FL on mobile clients but also can represent the data distribution of individual clients' local data. Hence to guide the personalized aggregation for each individual clients, We propose BS-pFL, which can achieve low cost personalized federated learning through using bit-streams to predict the significant weight gradients/partial sums and also to represent the sparse weight gradient pattern of each client. From our experiments, our proposed framework can improve the computation efficiency by 3.5× on average (up to 4.2×) and reduce the communication cost by 23% on average (up to 41%) while still achieving the state-of-the-art personalized accuracy.

REFERENCES

[1] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *CoRR*, vol. abs/1704.05021, 2017. [Online]. Available: http://arxiv.org/abs/1704.05021

[2] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," 2015.

[3] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *CoRR*, vol. abs/1912.00818, 2019. [Online]. Available: http://arxiv.org/abs/1912.00818

[4] P. Dai, J. Yang, X. Ye, X. Cheng, J. Luo, L. Song, Y. Chen, and W. Zhao, "Sparsetrain: Exploiting dataflow sparsity for efficient convolutional neural networks training," in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, ser. DAC '20. IEEE Press, 2020.

[5] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *CoRR*, vol. abs/2003.13461, 2020. [Online]. Available: https://arxiv.org/abs/2003.13461

[6] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," *CoRR*, vol. abs/2006.08848, 2020. [Online]. Available: https://arxiv.org/abs/2006.08848

[7] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," 2020.

[8] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning: A meta-learning approach," *CoRR*, vol. abs/2002.07948, 2020. [Online]. Available: https://arxiv.org/abs/2002.07948

[9] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1510.00149

[11] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *CoRR*, vol. abs/2002.05516, 2020. [Online]. Available: https://arxiv.org/abs/2002.05516

[12] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," 2017.

[13] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," 2017.

[14] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 389–400. [Online]. Available: https://doi.org/10.1145/3210240.3210337

[15] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," 2017.

[16] H. Lu and Q. Zhang, "Applications of deep convolutional neural network in computer vision," vol. 31, pp. 1–17, 01 2016.

[17] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," 2017.

[18] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *CoRR*, vol. abs/2002.10619, 2020. [Online]. Available: https://arxiv.org/abs/2002.10619

[19] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," 2017.

[20] Q. Wan and X. Fu, "Fast-bcnn: Massive neuron skipping in bayesian convolutional neural networks," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 229–240.

[21] W. Wen, Y. He, S. Rajbhandari, M. Zhang, W. Wang, F. Liu, B. Hu, Y. Chen, and H. Li, "Learning intrinsic sparse structures within long short-term memory," 2018.

[22] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li, "Coordinating filters for faster deep neural networks," 2017.

[23] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1508–1518.

[24] X. Ye, P. Dai, J. Luo, X. Guo, Y. Qi, J. Yang, and Y. Chen, "Accelerating cnn training by pruning activation gradients," 2020.

[25] J. Zhang, X. Chen, M. Song, and T. Li, "Eager pruning: Algorithm and architecture support for fast training of deep neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 292–303. [Online]. Available: https://doi.org/10.1145/3307650.3322263

[26] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," *CoRR*, vol. abs/2012.08565, 2020. [Online]. Available: https://arxiv.org/abs/2012.08565

[27] X. Zhang, F. Wu, and Z. Li, "Application of convolutional neural network to traditional data," *Expert Systems with Applications*, vol. 168, p. 114185, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420309192

[28] P. Zhou, X. Yuan, H. Xu, S. Yan, and J. Feng, "Efficient meta learning via minibatch proximal update," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.